이차 순환신경망에서 정규문법의 학습을 위한 최대 epoch 결정

정현기[†] · 정순호[††]

요 약

기존의 정규문법 학습에 사용된 이차순환신경망의 학습 알고리즘은 최대 epoch 설정에서 좀더 분석적이지 못하여 학습이 실패한 경우 학습비용의 낭비가 발생하고 학습효율이 낮아진다. 이 논문에서는 학습알고리즘의 미비점을 학습과정에서 발생 될 수 있는 상황을 통해 분석하여 적절한 최대 epoch를 정함으로써 학습효율을 향상시키고자 한다. 그러기 위해 정규문법의 학습과정에서 소요되는 계산시간과 최대 epoch의 학습비용함수를 이론적으로 표현하고 이에 따라서 최대 epoch가 400~500일 때 최소 비용을 갖게 됨을 보이고 이것을 실험을 통해 확인한다.

Maximum Epoch for Learning Improvement of Second-Order Recurrent Neural Network Inferring Regular Grammars

Jung Hyeon Ki[†] and Jung Soon Ho[††]

ABSTRACT

Learning algorithm of SRNN doesn't use analytic maximum epoch, so that its performance is inefficient and its cost is high. In this paper, with the proper maximum epoch, we improve learning efficiency. We first describe cost function of maximum epoch and computation time theoretically. Then, using it, we propose that maximum epoch must be between 400 and 500. Estimated maximum epoch is verified by experiment.

1. 서 론

문법의 예들로부터 Finite-State Automata를 추론하는 연구가 신경망(Neural Network)의 한 모델인 순환 신경망(Recurrent Neural Network)을 사용하여 진행되어 왔고, 이에 대응하는 여러 추론 방법들과 구조적 방법들이 학습 효과의 향상을 위해 개발되어 왔다[1-3,5,9-11,17]. 최근에 이들 연구에서는 학습성능이 좋은 이차 순환신경망(SRNN)을 모델로 사용하며[9], 학습 알고리즘은 비용의 낭비를 최소화한 개선된 Pseudo-gradient 방법을 사용한다[17]. 이 학습 알고리즘은 학습을 무한히 반복하여도 학습이 되지 않을 때 기존의 알고리즘이 학습을 계속 진행시킨데 반해 개선된 알고리즘은 그 원인을 알고리즘내의 함수를 통해 찾아내어 그 시점에서 학습을 중단하여 이후의 학습비용의 낭비를 줄인다[16,17].

이차 순환신경망을 이용해 문법을 학습할 때 개선된 알고리즘 역시 학습이 계속 진행되는 상황을 부분적으로는 알고리즘 내에서 해결하였지만 일부는 여전히 계속 학습을 하게되고, 따라서 학습을 종료시킬 수 있는 가장 효율적인 최대 epoch의 설정이 필요하다. 이 최대 epoch 값은 이전 연구에서는 경험적으로 제공하는데 만약 최대 epoch를 크게 설정한다면 학습이 계속 진행되는 상황에서 비용의 낭비가 발생하게 되고, 그렇지 않고 작은 값을 설정한다면 학습률이 떨어지는 결과가 발생한다.

† 부경대학교 전자계산학과 석사
†† 부경대학교 컴퓨터 멀티미디어 공학부 교수

적절한 최대 epoch를 설정하기 위해 학습률을 결정하는 반복횟수와 학습에 소요되는 계산시간을 비용함수를 통해 나타내어 최대 epoch를 결정하게 된다.

2장에서는 실험을 위해 사용된 이차 순환신경망 모델과 개선된 알고리즘에 관해서 소개하고, 3장에서는 학습률과 계산시간을 비용함수와 반복횟수를 통해 결정하게 된다. 그리고 4장에서는 실험을 통해 결과를 검정하고 마지막으로 5장에서는 결론을 언급하고자 한다.

2. 관련연구

이 장에서는 학습집합으로 사용되는 Tomita 문법과 실험에 사용된 모델인 이산 이차 순환신경망 (Discrete-SRNN)의 구조와 수정된 학습방법에 대해 설명한다.

2.1 Tomita Grammar

Tomita에 의해 1987년 Bench mark testing을 위해 소개된 정규문법으로 모두 7개의 문법으로 구성되어있다[4,6].

표 1. Tomita 문법의 정규표현

T_1 = 1^*
T_2 = $(10)^*$
T_3 = 짝수개의 연속적인 '0' 후에 짝수개의 연속적인 '1'을 가지는 스트링
T_4 = '000'을 포함하지 않는 스트링
T_5 = $[(01\|10)(01\|10)]^*$
T_6 = '1'의 개수에서 '0'의 개수를 뺀 결과를 3으로 나눈 나머지가 0인 스트링
T_7 = $0^*1^*0^*1^*$

2.2 개선된 SRNN

3개 상태 S_0, S_1, S_2를 가진 이산 이차 순환신경망은 입력인 0과 1에 의해 제어되는 두 개의 분리된 networks인 net0과 net1로 이루어져 있으며 그림 1과 같이 보여진다. time step t 에서 상태 S_i^t는 식(1)에 의해 식(2)과 같이 표현된다.

$$h_i^t = f(\sum_j w_{ij}^{x^t} S_j^{t-1}) \quad (1)$$

$$S_i^t = D(h_i^t) \quad (2)$$

여기서 f는 sigmoid function으로 $f(x) = \frac{1}{1+e^{-x}}$ 이고 D는 discrete function으로 $D(x) = \begin{cases} 0.8 & x \geq 0.5 \\ 0.2 & x < 0.5 \end{cases}$ 와 같다.

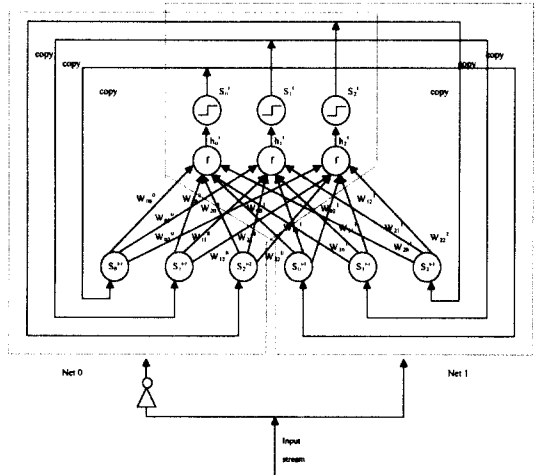



그림 1. 상태수가 3인 이산 이차 순환신경망

이런 discrete된 값을 사용하면 상태공간에서 몇 개의 값으로만 표현되어 불안정한 상태가 발생하지 않고 학습이 이루어진다. 그러나 analog 값 대신 discrete 된 값을 사용하므로 변화량의 기울기를 계산할 수 없다. 대신 그 값의 추정치를 사용하고 이를 Pseudo- gradient 라고 하고 아래와 같이 계산된다[1,7,8,12,14,15].

$$\frac{\partial \widetilde{h}_k^t}{\partial w_{ij}^n} = f' \cdot (\sum_l w_{kl}^{x^t} \frac{\partial \widetilde{h}_l^{t-1}}{\partial w_{ij}^n} + \delta_{ki}\delta_{nx^t} S_j^{t-1}) \quad (3)$$

이런 기존의 학습알고리즘은 학습과정에서 비용의 낭비가 발생하는데 그 이유는 에러가 허용치 보다 큼에도 불구하고 weight w_{ij}^n의 변화가 없게 되어 학습과정에 변화가 발생하지 않게 된다[17]. 이것은 식(3)에서 f' 값이 곱해지는데, f가 sigmoid 함수이므로 $f'(x) = f(x) \cdot (1 - f(x))$이다. 입력의 마지막 값인 L이 제공될 때 $f(L)$은 $f_{(L)} = f(\sum_j w_{0j}^{x^L} S_j^{L-1})$이 되고, 이 값이 0 또는 1이 될 때 $f'_{(L)} = f(\sum_j w_{0j}^{x^L} S_j^{L-1}) \cdot (1 - f(\sum_j w_{0j}^{x^L} S_j^{L-1})) = 0$ 이 되므로 weights수정이

불가능해진다. 위의 내용을 기존 알고리즘에 적용하면 다음과 같다.

□ 개선된 학습 알고리즘 □

```
select Tomita-Grammar
repeat up to MAXSEEDS seeds{
/* training phase */
  select initial weights
  repeat up to MAXEPOCHS epochs {
    success_string = 0;
    for n=1 to Training_Set_size {
      present string(n);
      pseudo_gradient_learning(n);
      evaluate error E;
      if ( E > error_tolerance ){
        compute f'(L) ;
      /* f'(L) = f'(∑_j w^{x,L}_{0j} S^{L-1}_j) */
        if ( f'(L) == 0 )
        break out of epochs loop; /* end epoch */
        compute gradient and weight update;
        change weights;
      }
      else increment success_string;
    }
  if (success_string == Training_Set_size)
   break out of seeds loop;
                    /*successful training*/
 } /* select new weights */
} /* training is failed */
```

여기서, epoch는 학습패턴을 모두 한번씩 학습했을 때의 계산시간이며 MAXEPOCH는 알고리즘에서 학습과정을 중단할 수 있는 유한한 값으로 설정한다.

기존의 알고리즘이 앞에서 언급한 문제점 때문에 학습이 되지 않는데도 불구하고 최대 허용 epoch까지 학습이 진행되는 시간적 낭비를 가지게 된다[13]. 개선된 알고리즘은 학습이 불가능해지는 시점을 f' 함수를 통해 알고리즘 내에서 찾아내어서 비용의 낭비를 막고 이 시점에서 새로운 weights를 다시 부여하여 학습이 모두 성공적으로 이루어지도록 한다[17].

3. 최대 epoch 설정

이 장에서는 문법의 학습에서 발생하는 상황과 최대 epoch를 설정하기 위하여 학습 알고리즘에서 사용하는 반복횟수와 학습시간에 사용되는 비용을 계산하기 위하여 정규문법의 학습상황을 살펴보고 그에 적합한 비용함수를 정의한다.

3.1 정규문법의 학습상황

정규문법을 개선된 학습알고리즘을 사용해 학습할 때 여러 상황이 발생하는데 그림 2는 정규문법의 학습과정에서 x축은 최대 epoch를 무한대로 설정했을 때 학습이 성공적인 실험횟수의 상대 빈도수를 y축으로 표현한 그래프이다. 이 그림에서 알 수 있듯이 학습의 성공적인 분포는 어느 구간동안 대부분 발생하고 이 후로 진행될수록 그 빈도수가 줄어든다. 그림 3은 그림 2의 최대 epoch를 무한대로 설정했을 때 실험횟수의 상대 빈도수를 누적된 값의 확률로써 나타낸 것으로 최대 epoch가 무한대를 가더라도 학습이 성공되지 못하는 학습횟수가 상당히 존재함을 나타낸다. 여기서 성공적인 학습이란 모든 학습패턴이 1 epoch동안 오차허용치(error tolerance)보다 적은 오차(error)를 가질 때를 말한다. 개선된 학습 알고리즘은 최대 epoch 설정을 통해 학습을 유한한 횟수만에 100%에 거의 근접하게 이루어지게 한다. 그 시점은 그림 3에서 M으로 표현되는 지점으로 학습과정의 비용을 최소화하기 위해 적절한 M값의 설정이 필요하다.

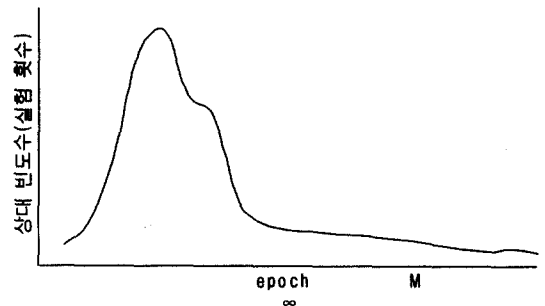



그림 2. 학습이 성공적인 분포 그래프

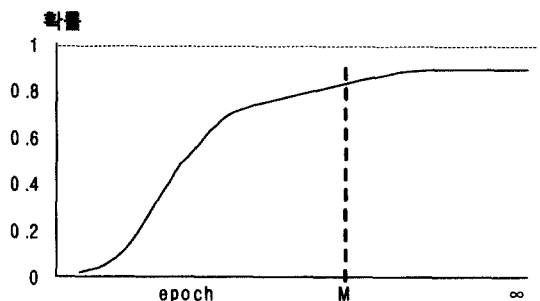



그림 3. 성공적인 학습의 누적 그래프

임의의 최대 epoch를 M으로 설정했을 때 학습 알고리즘의 결과는 아래와 같다.

(1) M epoch 이내에서 학습이 성공적인 경우
(2) M epoch 이내에서 학습이 불가능한 시점을 미리 찾아내는 경우
(3) M epoch에 도달하여 학습이 실패하는 경우

개선된 학습알고리즘은 위의 (2),(3)과 같은 경우에 새로운 가중치를 부여하여 학습을 다시 반복한다. 이럴 경우 알고리즘 내에서 새로운 학습이 가능한 횟수, 즉 반복횟수를 결정해 주어야 한다. 이 반복횟수가 클수록 학습률이 증가하는 반면 학습비용 역시 증가하여 둘 간의 trade-off 관계가 성립된다. 최대 epoch를 설정하는데 있어서 학습률과 관계되는 반복횟수와 학습에 사용된 계산시간인 비용을 고려하여 적절한 최대 epoch를 결정하게 된다.

3.2. 비용함수의 정의 및 반복횟수

최대 epoch를 결정하기 위하여 학습률에 따른 계산시간을 비용함수를 통해 평가하고 최소 비용을 가지는 epoch를 최대 epoch로서 결정하게 된다.

최대 epoch를 M으로 설정했을 때 1회 학습에서 성공할 확률 P_M은 임의의 최대 epoch인 M까지 성공적으로 학습된 실험회수의 누적 빈도수에 대한 확률이고, k는 반복횟수로 학습과정에서 학습이 성공하지 못한 경우는 알고리즘에 의해 새로이 학습을 시키게 되는데 이 때 새로이 학습이 가능한 횟수라고 하면 이 때 이 학습과정을 k번 반복할 때 총 학습 epoch의 기대값 $E(P_M, k)$를 비용함수라고 한다.

$$\begin{aligned} E(P_M, k) &= M + (1-P_M)^1 M + (1-P_M)^2 M \\ &\quad + \cdots + (1-P_M)^{k-1} M \\ &= M(1 + (1-P_M) + (1-P_M)^2 \\ &\quad + \cdots + (1-P_M)^{k-1}) \\ &= M\left(\frac{1-(1-P_M)^k}{P_M}\right) \quad (k > 0) \end{aligned} \tag{4}$$

반복횟수 k는 누적 학습률과 다음과 같은 관계가 있다.

누적 학습률 $= 1-(1-P_M)^k$.

반복횟수 k번 실행한 후 발생하는 학습실패율이 허용치인 ε보다 작을 때까지 학습을 반복하게 되는데 이 누적 학습률과 오차허용치간의 관계는

$$\begin{aligned} & 1-(1-P_M)^k > 1-\varepsilon \\ \Rightarrow\ & (1-P_M)^k < \varepsilon \\ \Rightarrow\ & k \log(1-P_M) < \log \varepsilon \end{aligned}$$

이다.

여기서 $(1-P_M) < 1$ 이므로

$$k > \frac{\log \varepsilon}{\log(1-P_M)} \text{ 이다.} \tag{5}$$

이러한 식(5)의 3차원 그래프는 그림 4와 같고 이것은 x축을 확률 P_M으로 y축을 허용치 ε로 했을 때 k값의 변화를 나타낸 것이다. 이 그림에서 확률이 적을수록 허용치 ε에 대한 k값은 증가하여 비용에도 영향을 미치게 된다. 그림 4는 허용치 ε일 때 확률 P_M을 보장하는 최소값이 곡면으로 나타나고 k값이 증가될수록 누적 학습률이 P_M보다 커짐을 나타낸다.

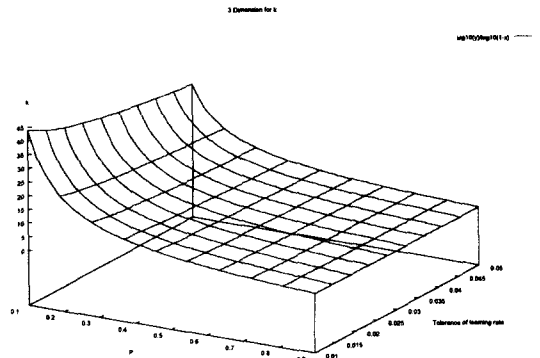



그림 4. P_M과 허용치 ε그리고 k의 관계 그래프

여기서 (4)와 (5) 식으로부터 비용함수의 값을 다음과 같이 정리할 수 있다.

$$E(P_M, k) > M\left(\frac{1-(1-P_M)^{\frac{\log \varepsilon}{\log(1-P_M)}}}{P_M}\right) \tag{6}$$

(6) 식에서 보는바와 같이 비용함수의 최저값은 오차 허용치와 확률 P_M의 함수로 표현될 수 있다. 그림 5는 오차 허용치 ε의 값이 0.05일 때 확률에 따른 최소비용을 보여주고 확률이 1일 때 가장 많은

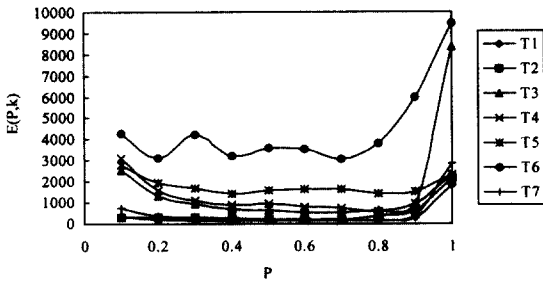



그림 5. (누적 학습률 ε = 0.05)

비용이 들며 확률이 중간정도에서 작은 값을 가짐을 보인다. 그림 6은 (6)번식의 함수를 이용하여 허용치 ε의 값을 0.01~0.05 까지 변화시켰을 때 전체 Tomita 문법에서 발생하는 학습비용을 평균값으로 계산하여 나타낸 것이다.

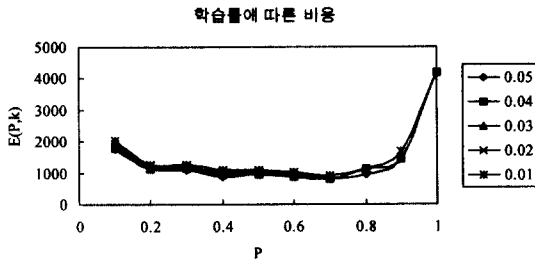



그림 6. 허용치 ε에 따른 평균 비용

표 2. 확률 P_M와 허용치 ε에 따른 비용

ε \ P_M	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0.05	1764	1111	1106	893	943	886	813	952	1436	4196
0.04	1813	1143	1157	893	943	886	813	1138	1436	4196
0.03	1877	1171	1157	959	1021	886	**813**	1138	1436	4196
0.02	1947	1220	1200	1012	1021	967	902	1138	1436	4196
0.01	2029	1275	1269	1090	1081	1022	902	1138	1677	4196

표 3. 확률 P에 따른 최대 epoch M

P_M	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
M	116	146	214	238	301	355	416	615	956	4196

이러한 상황을 구체적인 수치로 나타내면 표 2와 같고, 표 3은 확률에 따른 최대 epoch 값인 M값을 나타낸다. 이것을 확률 P_M과 허용치 ε, 비용함수 $E(P_M, k)$를 고려하여 3차원 형태의 그래프로 표현하면 그림 7과 같다.

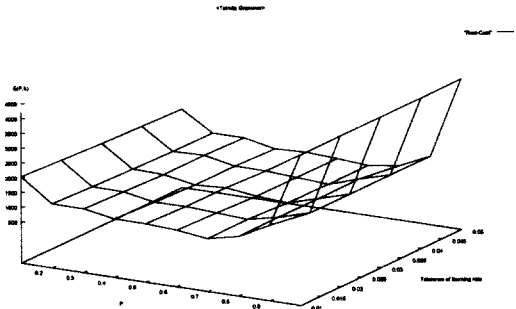



그림 7. Tomita 문법의 학습 비용

표 2와 그림 7을 통해서 알 수 있듯이 확률이 0.7인 시점이 학습비용이 가장 낮았고 반복횟수 k가 1,2,3일 때 최소 학습비용을 가지지만 k가 3일 때 다른 값에 비해 학습성공률이 97%이상으로 가장 높다. 그리고 이때의 M 값은 416 epoch를 가진다.

4. 실 험

Discrete SRNN에서 개선된 학습방법으로 Tomita 문법을 학습할 때 발생하는 3가지 상황을 보여주고, 실제 학습에서 사용되어진 시간을 고려하여 실험적으로 최대 epoch를 구하여 비용함수를 통한 결과와 비교한다.

Tomita 3번 문법에서 실제 실험횟수별로 학습되는 상황을 표시하면 그림 8과 같다.

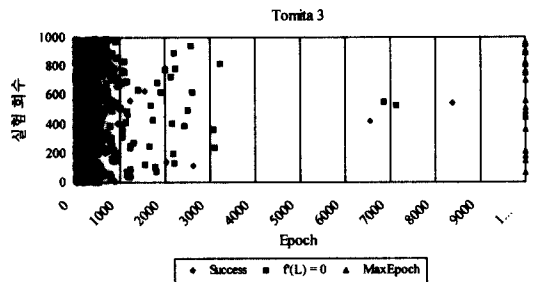



그림 8. 최대 epoch = 10000

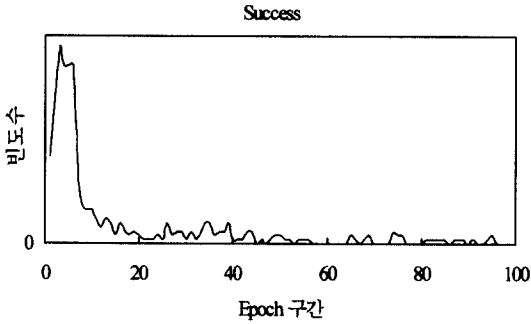
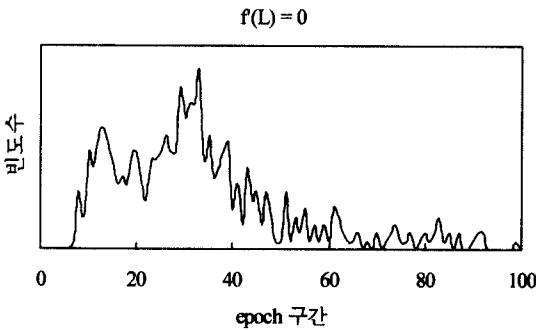
실제 최대 epoch를 아주 큰 값인 10000으로 했을 때 학습이 실패하는 경우 최대 epoch인 10000까지 학습이 진행되어서 비용의 낭비가 발생한다.

(a) 학습이 수렴된 분포

(b) f′(L) = 0에 의해 학습이 실패한 분포

그림 9. Tomita 3번의 분포 그래프

그림 9에서 대부분의 학습상황은 최대 epoch가 1000이하인 경우에서 발생하는 것을 알 수 있다. 이때 그림 9 (a)는 성공적인 학습의 실험횟수에 대한 빈도수의 분포를 나타내며, 그림 9의 (b) 학습과정에서 함수에 의해 학습이 불가능한 시점을 찾았을 때의 실험횟수에 대한 빈도수의 분포를 나타낸다. 나머지 Tomita 문법도 이와 유사한 결과를 가진다.

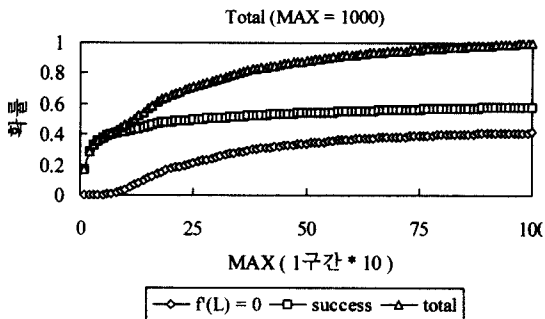



그림 10. 구간별 학습진행상황

위의 두 분포와 같이 모든 Tomita 문법을 고려하여 전체적인 흐름을 표현하면 그림 10과 같다.

그림 10과 같은 전체 분포를 고려하여 구간을 나눈 다음 구간별로 실험을 하여 실제 학습에 소요된 시간을 계산하면 그림 11과 같다.

아래 그림 12는 학습에 사용된 계산시간을 평균값으로 계산하여 표현한 것이다.

위의 실험을 통해 Tomita 문법 각각의 계산시간을 조사한 그래프에서 각 Tomita 문법에 따라 조금씩의 차이가 발견되지만 구간 200 ~ 700 사이에서 학습 실행시간이 다른 구간에 비해 적게 나왔다. Tomita 문법 전체를 모두 고려할 때 최대 epoch를 400으로 했을 때 가장 적은 시간이 사용되었다.

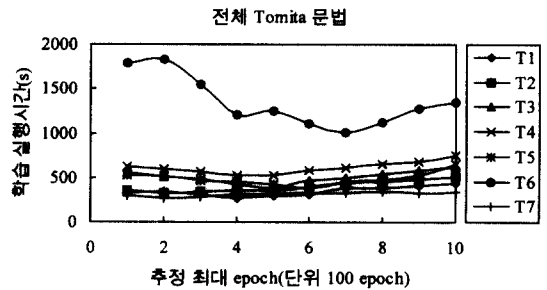



그림 11. Tomita 문법의 계산시간

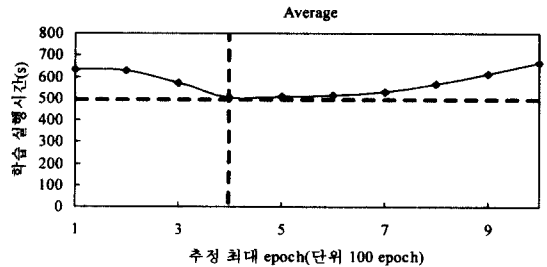



그림 12. Tomita 문법의 평균 계산시간

5. 결 론

이 논문에서는 이산 이차 순환신경망을 사용해 Tomita 문법을 학습할 때 기존 학습알고리즘의 문제점을 학습상황을 통해 분석하여 학습효율을 향상시키는 방법으로 적절한 최대 epoch값을 학습알고리즘에 적용하여 학습비용, 즉 실행속도를 줄인다.

최대 epoch를 설정하기 위해 이차 순환신경망을

이용한 Tomita 문법의 학습에서 성공적인 학습분포로부터 비용함수를 이론적으로 유도하여 정의한다. 이 함수를 이용하여 학습률에 따른 반복횟수를 결정하고 이를 문법의 학습에 적용하여 비용이 가장 적은 최대 epoch값을 결정한다. 그 결과 최대 epoch를 416으로, 반복횟수를 3으로 설정했을 때가 최소비용을 가진다. 실제적인 실험에서 400과 500 사이일 때 가장 작은 계산시간을 가진다. 이는 비용함수를 통한 결과와 거의 유사하여 가장 적절한 최대 epoch는 400과 500사이의 값으로 결정할 수 있고 반복횟수는 3회로 이때 학습률은 97%이상의 값을 가진다. 따라서 이를 이차 순환신경망을 사용한 정규문법의 학습에서 최대 epoch 값으로 제안하고 그 결과 학습효율을 향상시킬 수 있다.

참 고 문 헌

[1] Z. Zeng, R. M. Goodman, P. Symth, "Learning Finite State Machines With Self-Clustering Recurrent Networks", Neural Computation, VOL 5, pp.976-990, 1993.

[2] D. Angluin, "Inference of reversible languages", J.Assoc.Comput.Machin 29(3), pp.741-765, 1972.

[3] E. M. Gold, "System identification via state characterization", Automatics 8, pp.621-636, 1972.

[4] M. Tomita, "Dynamic construction of finite-state automata from examples using hill climbing", In Proceeding of the Fourth Annual Cognitive Science Conference, pp.105, 1982.

[5] A. Cleeremans, D. Servan-Schreiber, J. L. McClelland, "Finite state automata and simple recurrent networks", Neural Computation. 1, pp.372-381, 1989.

[6] J. L. Elman, "Distributed representations, simple recurrent network, and grammatical structure", Machine Learn, 7(2/3), pp.195-225, 1991.

[7] C. L. Giles, G. Z. Sun, H. H. Chen, Y. C. Lee, and D. Chen, "Second-order recurrent neural network", Neural Computation, 4(3), pp.393-405, 1992.

[8] D. Servan-Schreiber, A. Cleeremans, J. L. McCl elland, "Graded state machine; The represen- tation of temporal contingencies in simple recurrent network", Machine Learn, 7(2/3), pp.161-193, 1991.

[9] C. L. Giles, C. B. Miller, "The effect of higher order in recurrent neural network; experiments".

[10] C. L. Giles, C. B. Miller, D. Chen, H. H. Chen, G. Z. Sun, Y. C. Lee, "Learning and Extracting finite State Automata with Second-Order Recurrent Neural Networks", Neural Computation 4, pp.393-405, 1992.

[11] T. Lin, B. G. Horne, "Learning Long-Term Dependencies in NARX Recurrent Neural networks", IEEE Transaction on neural networks. VOL 7, NO 6, 1996.

[12] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult", IEEE Transaction on neural networks, VOL 5, NO 2, 1994.

[13] S. Muroga, "Lower Bounds of the Number of Threshold Functions and a Maximum weight", IEEE Transaction on electronic computation.

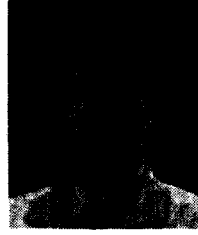
[14] Y. J. Lee, S. H. Oh, and M. W. Kim, "An analysis of Premature Saturation in Back Propagation Learning", Neural Networks, VOL 6, pp.719-728, 1993.

[15] P. Baldi, "Gradient Descent Learning Algorithm Overview: A General Dynamical Systems Perspective", IEEE Transaction on neural networks, VOL 6, NO 1, 1995.

[16] 류수길, 강효진, 정현기, 정순호, "유한 상태 오토마타의 추론을 위한 이차 순환 신경망의 학습시간 단축", 한국 멀티미디어 학회 '99 춘계 학술발표 논문집, pp.500-504, 1999

[17] 정현기, 정순호, "정규문법 추론을 위한 이산 순환 신경망의 개선된 학습방법", 한국정보처리학회 논문지 제 6권 12호 게재 예정

정 현 기

1996년 부경대학교 전자계산학과 학사
1999년 부경대학교 전자계산학과 석사
현재 부경대학교 전자계산학과 인공지능 연구실 연구원
관심분야 : 인공지능, 신경망, 컴퓨터 시각

정 순 호

1980년 서울대학교 수학교육과 졸업
1982년 한국과학기술원 전산학과 석사학위
현재 한국과학기술원 전산학과 박사과정
1982~1986 삼성전자 종합연구소 주임연구원
1986~1987 C & B Technology Co. 개발부장
1987~현재 부경대학교 컴퓨터 멀티미디어 공학부 교수
관심분야 : 인공지능, 신경회로망, 패턴인식, 기계학습, 컴퓨터 시각